

Chapter 6

Conclusion and Future Work

6.1 Summary of the results

6.2 Future work

6.2.1 wasm-mutate future work

Bibliography

- [1] Romano,A., Lehmann,D., Pradel,M., and Wang,W. (2022). Wobfuscator: Obfuscating javascript malware via opportunistic translation to webassembly. In *2022 2022 IEEE Symposium on Security and Privacy (SP) (SP)*, pages 1101–1116, Los Alamitos, CA, USA. IEEE Computer Society.
- [2] Narayan,S., Disselkoen,C., Moghimi,D., Cauligi,S., Johnson,E., Gang,Z., Vahldiek-Oberwagner,A., Sahita,R., Shacham,H., Tullsen,D., et al. (2021). Swivel: Hardening webassembly against spectre. In *USENIX Security Symposium*.
- [3] Lee,S., Kang,H., Jang,J., and Kang,B. B. (2021). Savior: Thwarting stack-based memory safety violations by randomizing stack layout. *IEEE Transactions on Dependable and Secure Computing*.
- [4] Johnson,E., Thien,D., Alhessi,Y., Narayan,S., Brown,F., Lerner,S., McMullen,T., Savage,S., and Stefan,D. (2021). Sfi safety for native-compiled wasm. *NDSS. Internet Society*.
- [5] Hilbig,A., Lehmann,D., and Pradel,M. (2021). An empirical study of real-world webassembly binaries: Security, languages, use cases. *Proceedings of the Web Conference 2021*.
- [6] Cabrera Arteaga,J., Laperdrix,P., Monperrus,M., and Baudry,B. (2021b). Multi-Variant Execution at the Edge. *arXiv e-prints*, page arXiv:2108.08125.
- [7] Cabrera Arteaga,J., Laperdrix,P., Monperrus,M., and Baudry,B. (2021a). Multi-Variant Execution at the Edge. *arXiv e-prints*, page arXiv:2108.08125.
- [8] Cabrera Arteaga,J., Floros,O., Vera Perez,O., Baudry,B., and Monperrus,M. (2021). Crow: code diversification for webassembly. In *MADWeb, NDSS 2021*.
- [9] (2021). Webassembly system interface.
- [10] Xu,Y., Solihin,Y., and Shen,X. (2020). Merr: Improving security of persistent memory objects via efficient memory exposure reduction and randomization. In *Proc. of ASPLOS*, pages 987–1000.

- [11] Tsoupidi,R. M., Lozano,R. C., and Baudry,B. (2020). Constraint-based software diversification for efficient mitigation of code-reuse attacks. *ArXiv*, abs/2007.08955.
- [12] Shillaker,S. and Pietzuch,P. (2020). Faasm: Lightweight isolation for efficient stateful serverless computing. In *USENIX Annual Technical Conference*, pages 419–433.
- [13] Runeson,P., Engström,E., and Storey,M.-A. (2020). *The Design Science Paradigm as a Frame for Empirical Software Engineering*, pages 127–147. Springer International Publishing, Cham.
- [14] Lehmann,D., Kinder,J., and Pradel,M. (2020). Everything old is new again: Binary security of webassembly. In *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association.
- [15] Harrand,N., Soto-Valero,C., Monperrus,M., and Baudry,B. (2020). Java decompiler diversity and its application to meta-decompilation. *Journal of Systems and Software*, 168:110645.
- [16] Gadepalli,P. K., McBride,S., Peach,G., Cherkasova,L., and Parmer,G. (2020). Sledge: A serverless-first, light-weight wasm runtime for the edge. In *Proceedings of the 21st International Middleware Conference*, page 265–279.
- [17] Chen,D. and W3C group (2020). WebAssembly documentation: Security. Accessed: 18 June 2020.
- [18] Bryant,D. (2020). Webassembly outside the browser: A new foundation for pervasive computing. In *Proc. of ICWE 2020*, pages 9–12.
- [19] Österlund,S., Koning,K., Olivier,P., Barbalace,A., Bos,H., and Giuffrida,C. (2019). kmvx: Detecting kernel information leaks with multi-variant execution. In *ASPLOS*.
- [20] Churchill,B., Padon,O., Sharma,R., and Aiken,A. (2019). Semantic program alignment for equivalence checking. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2019, page 1027–1040, New York, NY, USA. Association for Computing Machinery.
- [21] Aga,M. T. and Austin,T. (2019). Smokestack: thwarting dop attacks with runtime stack layout randomization. In *Proc. of CGO*, pages 26–36.
- [22] Silvanovich,N. (2018). The problems and promise of webassembly. Technical report.
- [23] Lu,K., Xu,M., Song,C., Kim,T., and Lee,W. (2018). Stopping memory disclosures via diversification and replicated execution. *IEEE Transactions on Dependable and Secure Computing*.

- [24] Li,J., Zhao,B., and Zhang,C. (2018). Fuzzing: a survey. *Cybersecurity*, 1(1):1–13.
- [25] Gupta,S., Saxena,A., Mahajan,A., and Bansal,S. (2018). Effective use of smt solvers for program equivalence checking through invariant-sketching and query-decomposition. In Beyersdorff,O. and Wintersteiger,C. M., editors, *Theory and Applications of Satisfiability Testing – SAT 2018*, pages 365–382, Cham. Springer International Publishing.
- [26] Genkin,D., Pachmanov,L., Tromer,E., and Yarom,Y. (2018). Drive-by key-extraction cache attacks from portable code. *IACR Cryptol. ePrint Arch.*, 2018:119.
- [27] Belleville,N., Couroussé,D., Heydemann,K., and Charles,H.-P. (2018). Automated software protection for the masses against side-channel attacks. *ACM Trans. Archit. Code Optim.*, 15(4).
- [28] WebAssembly Community Group (2017b). WebAssembly Specification.
- [29] WebAssembly Community Group (2017a). WebAssembly Roadmap.
- [30] Sasnauskas,R., Chen,Y., Collingbourne,P., Ketema,J., Lup,G., Taneja,J., and Regehr,J. (2017). Souper: A Synthesizing Superoptimizer. *arXiv preprint 1711.04422*.
- [31] Haas,A., Rossberg,A., Schuff,D. L., Schuff,D. L., Titzer,B. L., Holman,M., Gohman,D., Wagner,L., Zakai,A., and Bastien,J. F. (2017). Bringing the web up to speed with webassembly. *PLDI*.
- [32] Van Es,N., Nicolay,J., Stievenart,Q., D’Hondt,T., and De Roover,C. (2016). A performant scheme interpreter in asm.js. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, SAC ’16, page 1944–1951, New York, NY, USA. Association for Computing Machinery.
- [33] Couroussé,D., Barry,T., Robisson,B., Jaillon,P., Potin,O., and Lanet,J.-L. (2016). Runtime code polymorphism as a protection against side channel attacks. In *IFIP International Conference on Information Security Theory and Practice*, pages 136–152. Springer.
- [34] Morgan,T. D. and Morgan,J. W. (2015). Web timing attacks made practical. *Black Hat*.
- [35] Kim,D., Kwon,Y., Sumner,W. N., Zhang,X., and Xu,D. (2015). Dual execution for on the fly fine grained execution comparison. *SIGPLAN Not*.
- [36] Davi,L., Liebchen,C., Sadeghi,A.-R., Snow,K. Z., and Monrose,F. (2015). Isomeron: Code randomization resilient to (just-in-time) return-oriented programming. In *NDSS*.

- [37] Crane,S., Homescu,A., Brunthaler,S., Larsen,P., and Franz,M. (2015). Thwarting cache side-channel attacks through dynamic software diversity. In *NDSS*, pages 8–11.
- [38] Baudry,B. and Monperrus,M. (2015). The multiple facets of software diversity: Recent developments in year 2000 and beyond. *ACM Comput. Surv.*, 48(1).
- [39] Alon Zakai (2015). asm.js Speedups Everywhere.
- [40] Agosta,G., Barenghi,A., Pelosi,G., and Scandale,M. (2015). The MEET approach: Securing cryptographic embedded software against side channel attacks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(8):1320–1333.
- [41] Le,V., Afshari,M., and Su,Z. (2014). Compiler validation via equivalence modulo inputs. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '14, page 216–226.
- [42] Homescu,A., Neisius,S., Larsen,P., Brunthaler,S., and Franz,M. (2013). Profile-guided automated software diversity. In *Proceedings of the 2013 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, pages 1–11. IEEE.
- [43] Maurer,M. and Brumley,D. (2012). Tachyon: Tandem execution for efficient live patch testing. In *21st USENIX Security Symposium (USENIX Security 12)*, pages 617–630.
- [44] Jackson,T. (2012). *On the Design, Implications, and Effects of Implementing Software Diversity for Security*. PhD thesis, University of California, Irvine.
- [45] Cleemput,J. V., Coppens,B., and De Sutter,B. (2012). Compiler mitigations for time attacks on modern x86 processors. *ACM Trans. Archit. Code Optim.*, 8(4).
- [46] Sidiroglou-Douskos,S., Misailovic,S., Hoffmann,H., and Rinard,M. (2011). Managing performance vs. accuracy trade-offs with loop perforation. In *Proceedings of the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of Software Engineering*, ESEC/FSE '11, page 124–134, New York, NY, USA. Association for Computing Machinery.
- [47] Jackson,T., Salamat,B., Homescu,A., Manivannan,K., Wagner,G., Gal,A., Brunthaler,S., Wimmer,C., and Franz,M. (2011). Compiler-generated software diversity. In *Moving Target Defense*, pages 77–98. Springer.
- [48] Amarilli,A., Müller,S., Naccache,D., Page,D., Rauzy,P., and Tunstall,M. (2011). Can code polymorphism limit information leakage? In *IFIP International Workshop on Information Security Theory and Practices*, pages 1–21. Springer.

- [49] Chen,T. Y., Kuo,F.-C., Merkel,R. G., and Tse,T. H. (2010). Adaptive random testing: The art of test case diversity. *J. Syst. Softw.*, 83:60–66.
- [50] Salamat,B., Jackson,T., Gal,A., and Franz,M. (2009). Orchestra: intrusion detection using parallel execution and monitoring of program variants in user-space. In *Proceedings of the 4th ACM European conference on Computer systems*, pages 33–46.
- [51] Maia,M. D. A., Sobreira,V., Paixão,K. R., Amo,R. A. D., and Silva,I. R. (2008). Using a sequence alignment algorithm to identify specific and common code from execution traces. In *Proceedings of the 4th International Workshop on Program Comprehension through Dynamic Analysis (PCODA)*, pages 6–10.
- [52] Jacob,M., Jakubowski,M. H., Naldurg,P., Saw,C. W. N., and Venkatesan,R. (2008). The superdiversifier: Peephole individualization for software protection. In *International Workshop on Security*, pages 100–120. Springer.
- [53] de Moura,L. and Bjørner,N. (2008). Z3: An efficient smt solver. In Ramakrishnan,C. R. and Rehof,J., editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340, Berlin, Heidelberg. Springer Berlin Heidelberg.
- [54] Salamat,B., Gal,A., Jackson,T., Manivannan,K., Wagner,G., and Franz,M. (2007). Stopping buffer overflow attacks at run-time: Simultaneous multi-variant program execution on a multicore processor. Technical report, Technical Report 07-13, School of Information and Computer Sciences, UC Irvine.
- [55] Bruschi,D., Cavallaro,L., and Lanzi,A. (2007). Diversified process replicæ for defeating memory error exploits. In *Proc. of the Int. Performance, Computing, and Communications Conference*.
- [56] Younan,Y., Pozza,D., Piessens,F., and Joosen,W. (2006). Extended protection against stack smashing attacks without performance loss. In *2006 22nd Annual Computer Security Applications Conference (ACSAC'06)*, pages 429–438.
- [57] Cox,B., Evans,D., Filipi,A., Rowanhill,J., Hu,W., Davidson,J., Knight,J., Nguyen-Tuong,A., and Hiser,J. (2006). N-variant systems: a secretless framework for security through diversity. In *Proc. of USENIX Security Symposium*, USENIX-SS'06.
- [58] Pohl,K., Böckle,G., and Van Der Linden,F. (2005). *Software product line engineering: foundations, principles, and techniques*, volume 1. Springer.
- [59] Bhatkar,S., Sekar,R., and DuVarney,D. C. (2005). Efficient techniques for comprehensive protection from memory error exploits. In *Proceedings of the USENIX Security Symposium*, pages 271–286.

- [60] El-Khalil,R. and Keromytis,A. D. (2004). Hydan: Hiding information in program binaries. In Lopez,J., Qing,S., and Okamoto,E., editors, *Information and Communications Security*, pages 187–199, Berlin, Heidelberg. Springer Berlin Heidelberg.
- [61] Kc,G. S., Keromytis,A. D., and Prevelakis,V. (2003). Countering code-injection attacks with instruction-set randomization. In *Proc. of CCS*, pages 272–280.
- [62] Bhatkar,S., DuVarney,D. C., and Sekar,R. (2003). Address obfuscation: an efficient approach to combat a board range of memory error exploits. In *Proceedings of the USENIX Security Symposium*.
- [63] Barrantes,E. G., Ackley,D. H., Forrest,S., Palmer,T. S., Stefanovic,D., and Zovi,D. D. (2003). Randomized instruction set emulation to disrupt binary code injection attacks. In *Proc. CCS*, pages 281–289.
- [64] Chew,M. and Song,D. (2002). Mitigating buffer overflows by operating system randomization. Technical Report CS-02-197, Carnegie Mellon University.
- [65] Cohen,F. B. (1993). Operating system protection through program evolution. *Computers & Security*, 12(6):565–584.
- [66] Pettis,K. and Hansen,R. C. (1990). Profile guided code positioning. In *Proceedings of the ACM SIGPLAN 1990 Conference on Programming Language Design and Implementation, PLDI '90*, page 16–27, New York, NY, USA. Association for Computing Machinery.
- [67] Henry,M. (1987). Superoptimizer: a look at the smallest program. *ACM SIGARCH Computer Architecture News*, 15(5):122–126.
- [68] Avizienis and Kelly (1984). Fault tolerance by design diversity: Concepts and experiments. *Computer*, 17(8):67–80.
- [69] Ryder,B. G. (1979). Constructing the call graph of a program. *IEEE Transactions on Software Engineering*, (3):216–226.
- [70] Needleman,S. B. and Wunsch,C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. 48(3):443–453.
- [71] Gnanadesikan,R. and Wilk,M. B. (1968). Probability plotting methods for the analysis of data. *Biometrika*, 55(1):1–17.
- [72] Mann,H. B. and Whitney,D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *Ann. Math. Statist.*, 18(1):50–60.

Index

code replacement overlapping, 23

CROW , 18–24, 27, 31, 39, 40

MEWE , 18, 24, 27

multivariant , 12, 13, 18, 24–27, 29,
36–38, 45–47

program variant , 10–12, 23, 39

Software Diversification , 1, 8, 10, 12–
14, 21

Wasm , 4–9, 15, 18, 20, 22, 24, 27, 37

WebAssembly , 1, 3–5, 7, 8, 13–15,
17–20, 22, 24–26, 29–31, 35–
40, 43, 46